COSC 460

Research Project Report

Department of Computer Science

University of Canterbury

A Design and Implementation Study of Query-By-Example

Neroli Helen Anderson

October, 1982.

Acknowledgements

I would like to thank

Dr. R.E.M. Cooper, supervisor of this project

and Mr. D. Mackay, Computer Scientist,
Christchurch Clinical School of Medicine

for the help they have afforded me in this research project.

I would also like to acknowledge the use of computing facilities of the Christchurch Clinical School of Medicine.

N.H. Anderson

A Design and Implementation Study of Query-By-Example

Table of Contents

> "Computers have not arrived on the scene for aesthetic reasons, but because they are essential to the survival of a complex society" [10]

1. Introduction

Society today is becoming increasingly information-oriented and thus it is now important to determine means of organising data to maximise its utility. This requirement and the rapid expansion of computer technology have lead to the development of the concept of "a database". Defined simply, a database system may be viewed as "a computer system whose overall [function] is to record and maintain information" [2]. The nature of information is such that it will never be possible to define rigorously a structure to which every item of data will conform. This inherent complexity accounts for its usefulness to society and interest to Computer Scientists. A database will contain that information which is significant to the operation of the organisation in which the system functions. Further, this data will be structured in the manner most suitable to the application(s).

Data collection, storage, update and retrieval constitute the basic functions required of any information management system. The design of the methods of implementation of these operations is subject to a number of variable external factors such as source(s), structure, quantity, nature and users of the information.

Of these elementary functions this study will be concerned with the information retrieval component. A query language will, for purposes of this report, be defined as being "that part of the interface between human users of a computer database and the database system responsible for the retrieval of information". One of the most important characteristics of any query language is how well it is tailored to the problem area [4].

The application environment being studied in this report is a small medical database. The project is concerned with the aforesaid variables with respect to the design and implementation of a query language suitable for this database. This information system was established in 1982 for the Christchurch Clinical School of Medicine, Paediatric Department and contains raw data pertinent to proposed, but as yet undefined, research studies.

This report surveys and evaluates the commonly used query languages as they may be applied to the Paediatric Database. On the basis of this knowledge a retrieval system suitable for this environment has been designed.

"If you spend all your days talkin' to a machine,
You might forget that you're a human being "

from 'The Joy of Cooking'
Bear Brown Publishing Company, 1971 [4]

2. Query Languages

There is much discussion as to the nature, structure and definition of query languages. The commonly used implementation strategies are :-

* the language is rigorously defined according to a set of syntax rules. In accordance with this syntax the user must translate her query to an equivalent structure in the language
* the user generates a query for information exactly as conceived in her mind and then by iterative interactive refinement with the system it is translated to an unambiguous form acceptable to both user and machine
* menu approach requiring simple responses to a prestored set of options [4].

No one facility is likely to emerge as universal, although, the more easy to use and well designed it is, the greater will be its acceptance.

In this section of the study the term "user" will be employed in a general sense to signify any person lacking in a specialised computer background using the system with the intent of obtaining information from it.

2.1 Requirements and Expectations

2.1.1 User

It is necessary to consider those query language features facilitating the successful interaction of a user with the database. Perhaps the most important aspects are that the system be reliable and secure. Available literature on this subject also stresses that the system should display "ease of use" and "user friendliness". These expressions are subjective and difficult to define precisely. They are generally interpreted as meaning that the system must display attributes that will appeal to all categories of user. The definition of these features will be determined largely by the user environment. Considerations in determining such components include :-

(i) level and format of interactive dialogue required

(ii) format of output

(iii) level of help, error detection and correction facilities

(iv) user experience and familiarity with computer equipment

(v) response time.

The user should expect the freedom to generate any desired query and be confident that the query asked would be answered, not some abstraction of it.

The syntax needs to be be clearly defined to avoid problems of ambiguity and confusion.

The system should be able to understand and perform the necessary translations between the structure of the data as understood by the user and the physical representation of it.

In addition, the user should be assured of the consistency of the translation process.

2.1.2 System

The preceding list suggests those factors in a query language desirable to users of the system. Similarly the computer system has certain requirements that must be adhered to in the design of the language.

Perhaps the most important consideration in this respect is that the execution of the query language should not interfere with the operation of the computer system. That is, performance of other system functions should not be adversely affected. Also, there should be no posssibility of a threat to the security of the system. An optimisation, suitable to the structure of the computer system, of code, memory and storage references and the operation of peripheral devices is most desirable.

2.2 Design Considerations

On the basis of the established desirable system criteria the design considerations to satisfy these should be studied.

Section 2.1.1 considered the user requirements of a database system. The design of the system should aim to satisfy these demands. The most difficult to define and satisfy is the idea of a "user friendly" system. Many factors influencing this will be peculiar to the environment. That is, they may be considered as being subject to "user type". The determination of this variable should be the result of a study considering at least the following points :-

(i) user education - experience and attitudes

(ii) main user requirements and reasons for using the system

(iii) user intelligence

(iv) frequency of use, length of sessions and method of query generation.

System design must also be very concerned with the language definition. An optimal size and structure should be achieved, sufficient to guarantee power yet not to discourage usage. The syntactic rules should aim to satisfy the following criteria :-

(i) lessen the possibility of ambiguity and user error

(ii) avoid confusion between the language and other formal language systems (eg. mathematics, statistics, English)

(iii) force consistency of meaning and interpretation

(iv) facilitate expansion and modification subject to the projected useful life of the database.

An excellent approach, where possible, to aid in the planning of a system, is to identify the "user type" and the nature of proposed queries. This combination will facilitate the design of the system and language as an integrated unit.

In addition to the above it must be anticipated that the user requirements are likely to be subject to continual change as the power of the language is appreciated and the specifications modified.

2.3 Database Users

At this stage in the study it is appropriate to define more precisely the term "user of a database system". Available literature [3b,4] proposes that there exist three major classes of user requiring access to the information stored in a database :- application programmers, technical personnel/non programmer professionals and casual users. The following definitions of users, in accordance with the above proposals, will be adopted for the remainder of this report.

Application Programmer

one assumed to have a knowledge of computer programming languages and concepts and with an understanding of the organisation of the data but often able to ignore its meaning.

Non-programmer Professional

one who specialises in the semantics of the data [4a] but who does not wish to be concerned with the computer system details.

Casual User

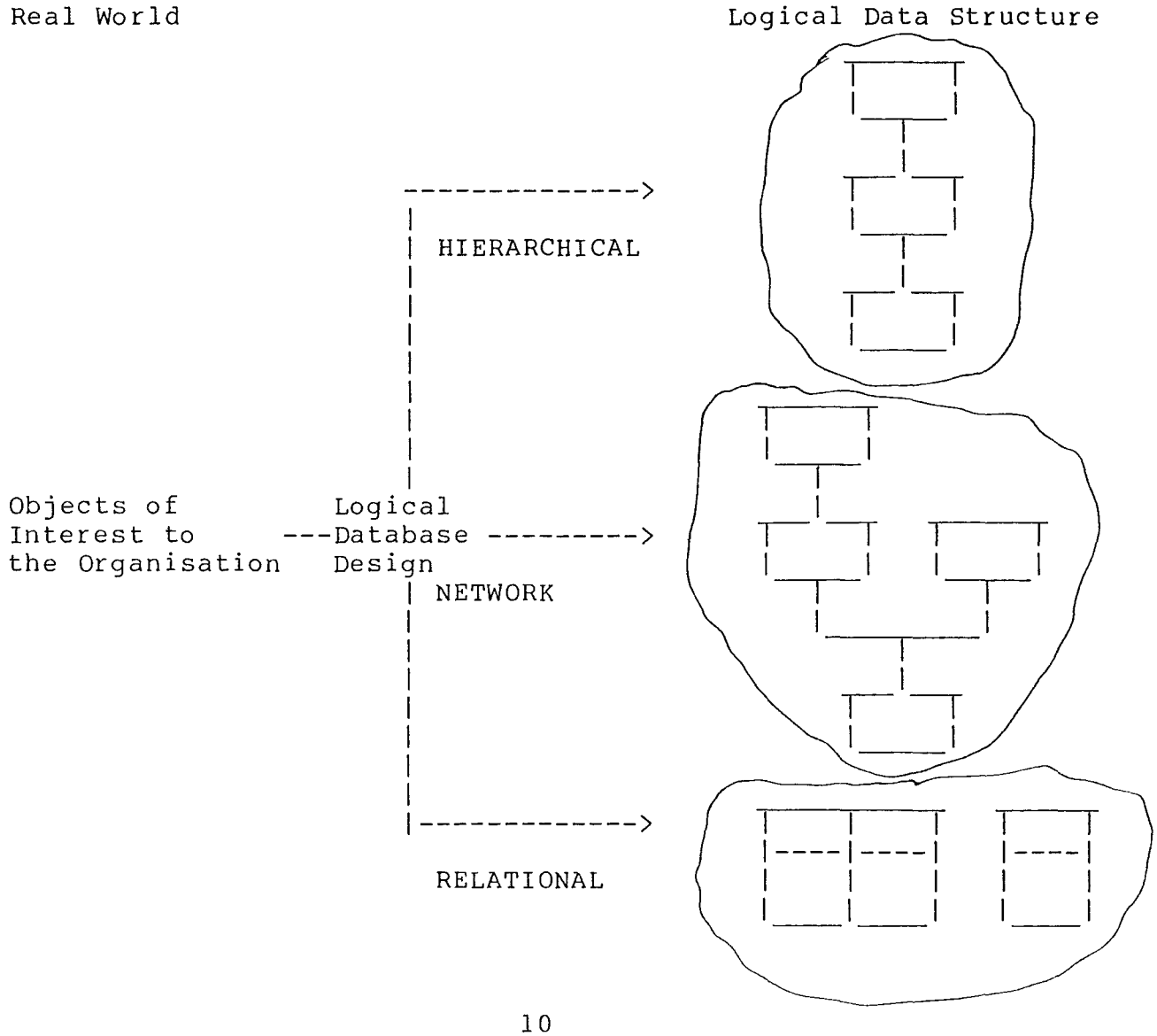
this user type has been defined by E.F. Codd [3b] to be "one whose interactions with the system are irregular in time and not motivated by job or social role. Such a user cannot be expected to be knowledgable about computers, programming, logic or relations".

The first category, application programmers, will be ignored for the pupose of this study. The system under consideration is devoid of any such personnel.

2.4 Database Architecture

The architecture of the database is also an important consideration - that is, does it conform to one of the traditional hierarchical, relational, network approaches, a derivation of these or is it unique ? Recently, the approach known as the "Entity-Relationship" model has developed as a technique to aid in the design of the logical structure of databases. For the purposes of design it may be helpful to abstract the database structure to this conceptual level.

The following figure illustrates these models [Fig. 11, 12].



The design of the Paediatric database resembles that of a relational model.

This report will be futher restricted by considering only those query languages that have been implemented for relational model databases. Techniques, where applicable, used in the other models will be referred to as necessary. Further, languages which form units in large system database packages (eg INQUIRY of Burrough's DMSII) will not be evaluated as the magnitude of such systems has little in common with the Paediatric database.

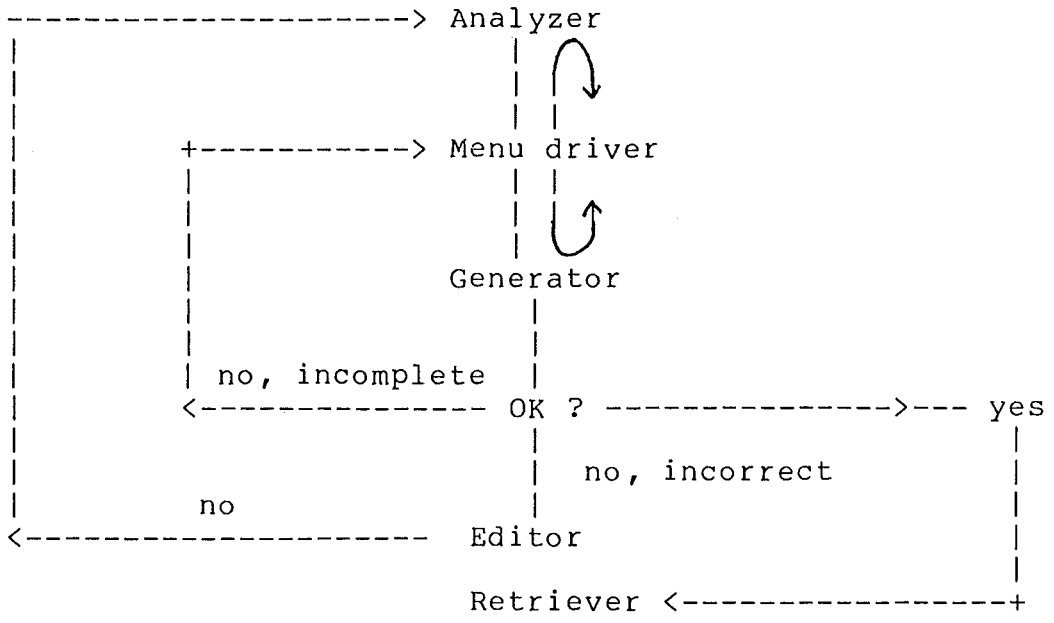
2.5 Query Language Design Models

Previous mention has been made of the diversity of opinion with regard to the design and implementation of query languages. The remainder of this chapter will be devoted to an overview of current implementations of query languages for relational model databases.

2.5.1. Natural Language

Codd and others argue that "the only way to entice casual users to interact with a computerised database is to permit them free use of their native tongue "[3a]. To this end he proposed Rendezvous Version 1. Codd in his definition of this model states "to have any hope of being viable, a natural language query system must 'talk' to the user using the same natural language as [she] uses except that its style has to be much more precise than that used by the average user" [3a].

This language he describes by the following diagram :-



[3a]

Rendezvous exemplifies the main design features of natural language systems. Perhaps the most important characteristic of this type of language is that the user is permitted almost unlimited freedom subject to a minimum number of formal rules.

Any natural English system must place restrictions on vocabulary and syntax. It may be difficult for a user to keep in mind such restrictions especially when daily conversations may induce forgetfulness [7]. Hence the margin for this, and other user errors is high. As a consequence of this freedom the possiblity also arises of user frustration with the repetitive interactive dialogue to formulate the query.

It is known that large complex program code and file structuring will be required to implement any natural language system. The cost of the dialogue with this system would be high and the time to obtain user verification would be significant.

2.5.2 Formal Languages

These languages have been defined to lie in the range between formal programming languages and natural English. There are several successful implementations of such languages available :-

(i) SQUARE developed by IBM for the System-R Database Management System. This language is based on relational algebra and has evolved into

(ii) SEQUEL a stand-alone language that does not need to be imbedded in any programming language. It is considered to lie between relational calculus and algebra. This classification is a result of its development from the algebraic language SQUARE and its features which are not not unlike those of calculus.

Example :-

```
select name
from   members t
where  10 <=
            select sum(quantity)
            from   orders
            where  name = t.name
```

print the names of all members who have ordered 10 or more pounds of food [9].

(iv) QUEL developed for the INGRES system at the University of California, Berkley. This language is based on relational calculus

Example :-

```
range of t is SUPPLIERS
retrieve (t.SNAME, t.SADDRESS)
```

[11]

print the names and addresses of all suppliers.

A language of this type forces the user to obey a formal notation and syntactic rules. It is however, able to retain an English-like quality. The need for discourse to resolve the query, essential to a natural language system, is lessened. Studies have shown that this model appeals more to the specialist or frequent user or those of a scientific calibre. This hypothesis is substantiated by a study undertaken by Reisner, Boyce and Chamberlin [4b].

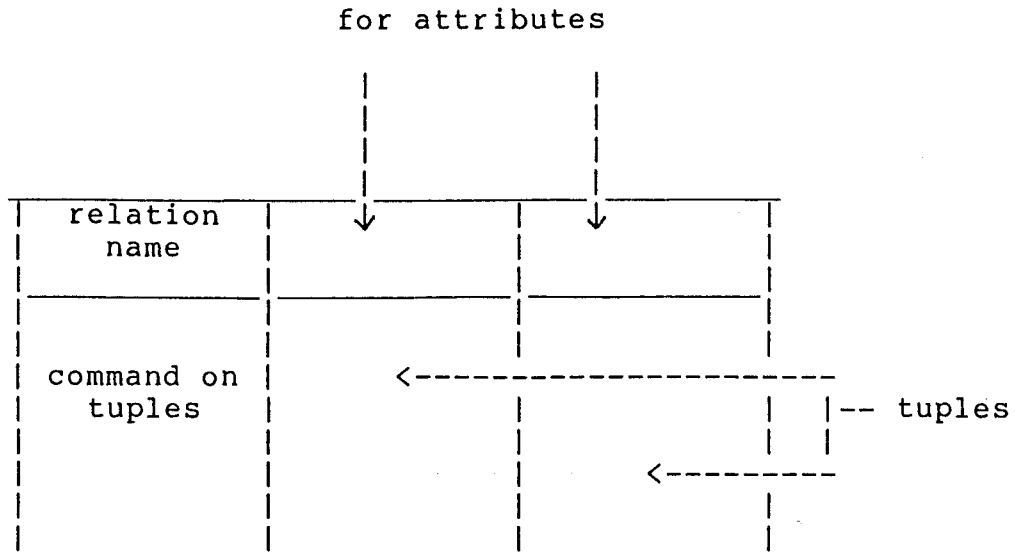
The complexity and magnitude of the software for this system would be predicted to be less than that required for natural language, but of considerable size.

2.5.3 Query-By-Example

Query-by-example (QBE) was developed by M. Zloof at IBM. It is designed to operate on a relational database and is known as a domain calculus language [9].

This language is specifically designed to be used sitting at a terminal. It is described as having "two dimensional syntax" since the query is not represented in the traditional linear fashion but as a skeleton table filled in as appropriate by the user.

Queries are posed using domain variables, constants and functions. The concept of QBE is illustrated by the table over



Prior to the creation of a query the user "calls up" the desired table to the screen. Using special cursor controls the appropriate values are entered in the columns of the table to generate the query. The function "P." found in any field indicates that if the query is satisfied then the value stored in the database corresponding to that column type, is to be output as a result. A simple example such as :-

"print the names of all the people in COSC who attended CGHS" would be represented thus :-

EXAMPLE	NAME	SUBJECT	SCHOOL
	P._x	COSC	CGHS

This illustration exemplifies the concepts of

(i) example (denoted by underscore). An example is

necessary when linking is required between rows of a table. The actual text of the example is meaningless and could just have as correctly been, say, 'P._salmon'.

(ii) constant. In order for the query to be satisfied by a logical database record the record field corresponding to the column heading must have the same value as (or satisfy some qualification on eg. <, >, <=) the constant.

(iii) functions, 'P.", the print function.

Any row containing the print function generates a new output table. The output for the preceding example would resemble :-

NAME
NHA
SJI

A table may consist of one or more rows, known as tuples. For a tuple to be satisfied it is necessary for all the columns of it to be fulfilled by the appropriate fields in the database record. Given that each individual tuple is satisfied, the context of the query will determine whether the tuples of the relation should be logically 'and'-ed or 'or'-ed to produce the final logical result (qv. Appendix D).

Clearly, QBE is significantly different to the previously illustrated query languages although still able to retain the attribute of being highly user oriented. This is achieved, though, through a shift of emphasis from the traditional linear

representation to the tabular approach. Consequently a powerful, yet simple to use, language has developed. Syntactic and exception rules are minimal. A query will be valid provided the attributes and constants are known to the system and the simple syntax is obeyed.

QBE is well suited to the user trained in a formal scientific discipline, that is, those of the classification "non-programmer professional". The user "needs to study the system only to that point of complexity which is compatible with the level of sophistication required within the domain of the query" [3b]. She is afforded great flexibility as the order of tuple formulation is immaterial.

As a consequence of the above features and the stand-alone ability of the language, QBE should therefore not require an unreasonable amount of program code.

The studies of D. Greenblatt & J. Waxman [3c] and J.C. Thomas & J.D. Gould [7] have yielded very favourable results concerning user reaction to QBE. The results of these two studies are claimed by Greenblatt and Waxman to "show the superiority of QBE as a database query language " [7].

A final comment that should be made with respect to QBE and its possible application to the Paediatric database is drawn from Thomas and Gould's study. That is, QBE is very useful to non-computer professionals, who desire to use the computer in a flexible and powerful manner on applications they know about, but without the necessity to spend a large amount of time learning to use the system.

3. PAEDIATRIC Database

The previous chapter considered query language design criteria. This section of the study shall be concerned with the Paediatric database and the environment into which these features must be made to apply.

3.1 Physical Environment

The database is established on a Systems Group System 2800 microcomputer. The hardware on which it operates consists of :-

64 K memory

Televideo 910 visual display unit (9600 baud, black and white display, 24 lines * 80 characters)

Diablo 630 daisywheel character printer

2 * 8 inch single sided double density floppy disk drives

It is implemented in an interpreted version of PASCAL, PASCAL-M, and a few assembly language routines. The system runs under the CP/M Operating System.

3.2 Users

The database system is established in a ward of Christchurch Public Hospital under the auspices of the Department of Paediatrics. The principal user in 1982 will be a research doctor, who was also responsible for the input specifications. At the conclusion of this academic year he will be leaving the department. The system will, however, be used by another doctor performing research of a similar nature.

The work undertaken by these doctors is largely the collection and analysis of patient data. At present this is all performed manually, a large part of the collating being done by a research nurse. Severe difficulties have been experienced in communicating to these persons the long-term potential of a database system.

This user group has therefore been classified as belonging to the "non-programming professionals" category. The database contains data for studies relating to the diseases Whooping Cough, Gastro-enteritis, Urinary Tract Infection and Bacterial Meningitis.

For the purpose of this report the users were able to provide a list of queries that would be a representation of the type of future enquiries (qv. appendix C). Reference to these will indicate the narrowness of their search requirements - most are merely numeric record counts and hence not utilising the full potential of the database.

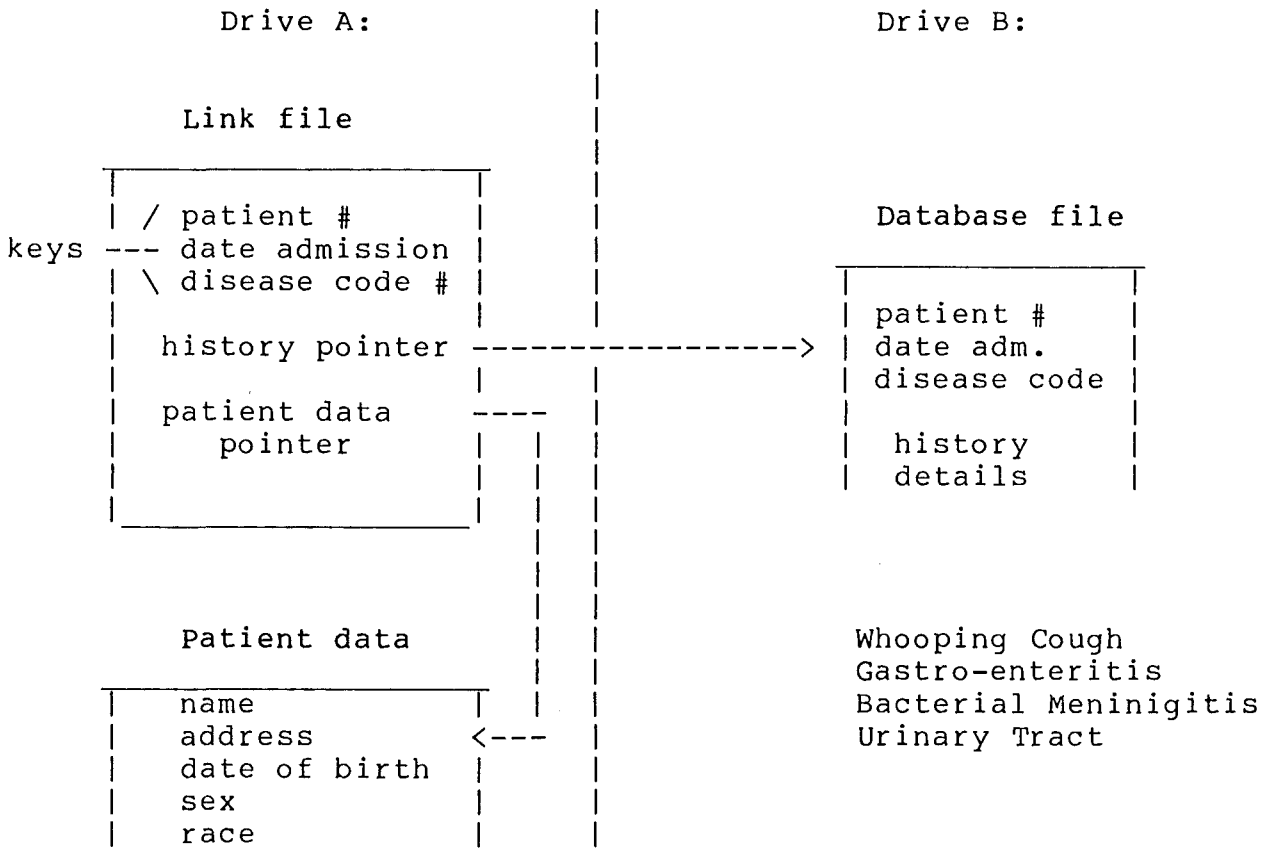
An extract from the User Manual indicating the data items stored with respect to Whooping Cough may be found in Appendix A. It is upon the users' knowledge of these input specifications that the sample queries were derived and the retrieval system must be based. It is envisaged that similar queries will arise for the other three diseases.

3.3 Database Architecture

The database storage system is run as four separate programs. This method was necessitated due to the amount of stack space available being significantly less than that required

for the successful compilation of the system as one program.

The database is distributed between a main disk and four "disease" disks thus :-



The design of the database requires that the combination patient number, date admission and disease identify uniquely a patient case history. As these keys are stored on both the link and the disease files the database is therefore able to be searched on either a patient or disease basis.

This implementation is a long-term plan and it is impossible to predict the future staff or their requirements. Time is not

an important constraint on the query language system design but, rather, a thorough systems analysis and the development of a useful design.

4. Design Model

The query language on which it was chosen to model the Paediatric retrieval facility is Query-by-example. The highly favourable aspects of QBE (qv. section 2.5.3) have already been enumerated. Of these and of specific interest to this environment are its abstraction from English and the tabular representation. As the potential users are familiar with analytical techniques, it will not be "unnatural" for them to converse in such a language but, rather, it will very much parallel their everyday conversations.

In addition to the above other reasons for this design choice included :-

(i) QBE requires a minimum knowledge of and familiarity with computer and peripheral hardware

(ii) the intelligence of the users suggests that they will exhibit quite short training times and be able to fully utilise the retrieval potentials

(iii) QBE, by design, affords the necessary flexibility for ad-hoc query generation and allows for easy expansion

(iv) the queries supplied were very easily able to be translated into QBE tables

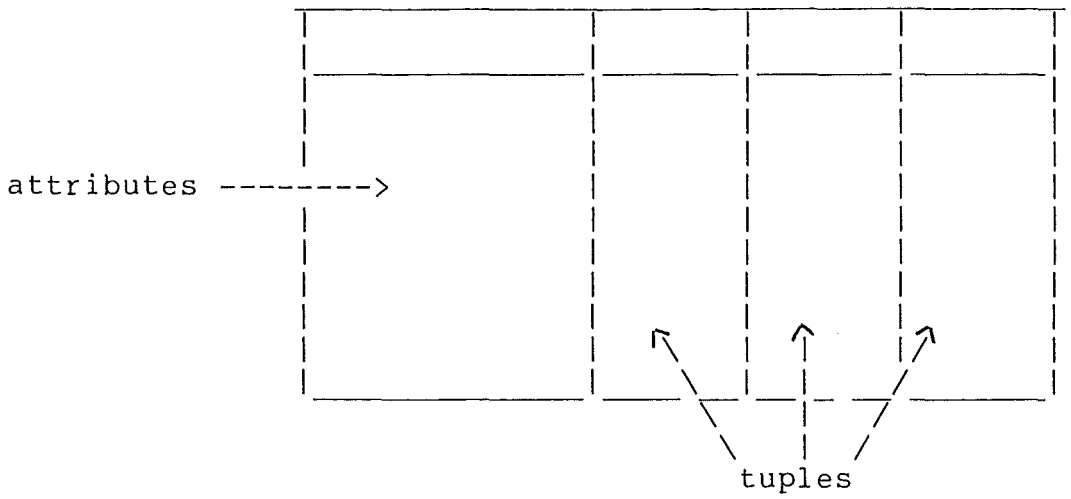
(v) the size of the code should be able to be successfully compiled and executed on this computer.

5. Implementation

In accordance with the previously enumerated considerations it is appropriate to consider the system as implemented.

5.1 External Features

The query table differs from the IBM portrayal as it has been rotated anticlockwise through a right angle. It occupies the entire screen thus :-



As a result of the users not being accustomed to the operation of computer equipment cursor movement is accomplished by the use of "arrow" keys.

Due to the specialist nature of the users detailed interaction with the program was considered to be unnecessary. Thus one only initial menu option is offered to ensure that the user is using the correct database disk and retrieval program. It is anticipated that although initially a greater level of assistance would be desirable the intelligence of the users is such that it will become redundant quickly .

5.2 Software Features

The syntax of the Paediatric Database QBE language may be found in Appendix B. In this initial implementation the language is very simple. This simplification was chosen for a number of reasons :-

(i) it will initially allow users to use the system with a minimum number of rules and thereby encourage confidence and usage. As the complexity increases so the users will more readily accept it

(ii) the indeterminate quantity of code the machine will successfully compile will force the query language to increase in complexity until the desired level of sophistication is attained or until it is impossible to compile the PASCAL program

(iii) as an aid to debugging

(iv) the user requirements are very vague and it appears futile to implement complex QBE techniques which may be totally unsuited to the environment.

The PASCAL program structure consists of the following modules :-

table input and syntax analysis (coded)

retrieval operations (coded)

output (uncoded).

Syntax analysis is performed as the query is entered. Following the successful generation of a query the retrieval procedure searches all the appropriate database records locating those which satisfy the the query. Case histories which fulfil

the requirements cause the appropriate output functions to be updated accordingly. At the conclusion of this process suitable output tables are generated and written to the user chosen output device(s).

6. Discussion

6.1 Differences

QBE is marketed as a software package by IBM. The Paediatric Database Query Language is modelled on this system but clearly it is neither suitable nor feasible to implement a faithful reproduction of it.

The Paediatric Database is designed to be a longterm project with, at present, no defined objectives. Consequently the scope of this QBE facility has been severely limited.

(i) It is assumed that tables (queries) are created on an ad-hoc basis. Also, since the system is very restricted in the amount of on-line storage space available, it would seem futile and wasteful to store the definitions of many possible tables on the off-chance of a stored table being required. The program code would also be greatly increased and as a result would not compile if both functions, recall and dynamic creation , were to be implemented. If only the recall option were adopted many desirable user objectives would be sacrificed.

(ii) The nature of the database is such that it is unnecessary to update/delete records, except in the case of erroneous initial entry. Thus, also dispensed with are the IBM options of update and deletion.

(iii) As a result of the predicted low probability of queries requiring linkages between tables, condition or command boxes these IBM options are not implemented. The latter

two facilities would not prove to be difficult extensions to the system but the need for them is doubtful.

(iv) Some of functions proposed by IBM are not suitable to this system ie. count unique, all (for the inclusion/exclusion duplicates) but others which would enhance operation do not exist. Such extensions to the range of functions provided could include variance, mean, differences, date calculations and age calculations.

(v) This system supports the linking of rows by specifying the same example within a column. IBM propose this method of linking within the entire query. It would not be a difficult or unreasonable extension to the Paediatric QBE to introduce this feature.

(vi) Several of the more complex IBM features - such as qualification of examples, the "dot" notation to indicate at least inclusion - were not implemented. In this initial design they were omitted as they have a level of complexity far greater than, it is envisaged, this database will require. Usage and requirements will indicate the necessity for their inclusion.

6.2 Extensions

It is appropriate to consider those features which have not been implemented but which would enhance the operation of this retrieval system. The imposed restrictions on the Paediatric QBE facility, specifically the (re)definition of suitable functions,

column linkages, condition and command boxes, are areas in which it may be desirable to extend Paediatric QBE facilities. Other such areas are :-

(i) the storage of common partial tables which are then extended in the session

(ii) the ability to negate an entire tuple

(iii) the acceptance of more than one function (in addition to print) to each table entry.

6.3 Suitability

The Paediatric database storage and retrieval systems have sought to meet the vague requirements of a very specialised user environment. Severe difficulties have hindered the development of the retrieval system to a working state. This lack of complete implementation should not be regarded as indicating the unfinished nature of the project. Rather, the main objective has been the language and system design.

The logic and data structures necessary for the the system to be brought to a working state have been finalised. To achieve this it was necessary to gain an in-depth understanding of the database structure, machine operation and capabilities and user requirements. This latter task proved to be most difficult.

By definition, non-programmer professionals cannot be expected to appreciate the subtleties involved in the design of system. The onus is therefore placed on the implementor to gain an understanding of the organisation. A significant amount has been learnt about this task. Commonly encountered difficulties between designer and users included :-

(i) due to the non-computer professional's lack of understanding of computer capabilities their specifications tend to have idealistic requirements.

(ii) a "communication breakdown" due to illegible abbreviations and terminology

(iii) the users' lack of understanding and interest in the concept of a database and its potential benefit to the organisation.

(iv) lack of coordination between the users and the lack of defined person with overall responsibility or interest.

It is felt that with proper training and an interest developed by the users that this system will be well suited to the environment.

7. Conclusion

The design of any information management system must be the result of a thorough evaluation of all aspects of the environment in which it must function.

This project has undertaken such a study for the Paediatric medical database. It has evaluated three information retrieval strategies as models on which to base a query language for the system. In conjunction with this knowledge, the requirements and constraints imposed, a relatively new and innovative technique, Query-By-Example, was chosen and adapted to the situation.

It is felt that a successful model and implementation strategy has been defined and that the Paediatric database will prove to be an asset with unlimited potential to the work undertaken by the staff of this department.

8. Appendices

Appendix A

This appendix consists of an extract from the Users' Manual, with questions pertaining to Whooping Cough. It is upon the users' knowledge of these that the sample queries (qv. Appendix C) were devised and that the retrieval system must be based.

Some questions have options available and there is an integer associated with each option. The answer(s) chosen is indicated by the user entering this number, which is then stored in a PASCAL set of the appropriate size. The user must choose at least one option. Questions referring to a "table" indicate the presence of a common data table (file) which is able to be called to the screen at input time as a "HELP" facility. Each item displayed has an integer (the record number) associated with it. It is this number that is entered by the user and stored in the database.

4. Questions

GENERAL

Patient number	(3 letters, 1 number)
date of admission	(dd/mm/yy)
name	(30 characters)
street address	(20 characters)
region	(qv REGIONS table)
sex	
race	(Asian/European/Maori/Polynesian/Other)
date of birth	(dd/mm/yy)

Infectious Contact

- within family

1. parent
2. sibling
3. peer
4. other

- outside family

5. playcentre
6. kindergarten
7. school
8. hospital
9. other

Social History

Parental Status

1. single	2. divorced
3. defacto	4. married
5. separated	6. widowed

Parental Occupation(s)

7. professional	8. managerial
9. skilled	10. clerical
11. semi-skilled	12. unskilled
13. umemployed	14. DPB
15. sickness benefit	16. other

Parental Race - Father

17. Asian	18. European
19. Maori	20. Polynesian
21. other	

Parental Race - Mother

22. Asian	23. European
24. Maori	25. Polynesian
26. other	

Parental Qualifications - Father

27. none	28. School Certificate
29. University Entrance	30. Bursary (7th form)
31. Tertiary	32. Trade Certificates
33. other	

Parental Qualifications - Mother

34. none	35. School Certificate
36. University Entrance	37. Bursary (7th form)
38. Tertiary	39. Trade Certificates
40. other	

Immunisation (relative to age)

41. nil	42. partial
43. complete	

Parental age - Father (enter 0 if unknown)
Parental age - Mother (enter 0 if unknown)
Family size
region in which family resides (as answered previously)

Referral Agency

1. GP
2. locum
3. A & E
4. Paediatrican
5. other

Antibiotic Therapy
- before admission (qv table ANTIBIOTIC)

Other relevant drugs
- before admission (qv table DRUGS)

Diagnosis
gram stain disease diagnosis (qv table DIAGNOSIS)
antigen disease diagnosis (qv table DIAGNOSIS)
culture disease diagnosis (qv table DIAGNOSIS)

Treatment 1

(i) antibiotic given
(ii) antibiotic analysis
- appropriate/inappropriate [A/I]
(iii) antibiotic dosage (mg/kg/day)
(iv) dosage analysis
- appropriate/inappropriate [A/I]
(v) frequency of administration (hours)
(vi) frequency administration analysis
- appropriate/inappropriate [A/I]
(vii) Duration of administration (days)
(viii) duration analysis
- appropriate/inappropriate [A/I]
(ix) route - appropriate/inappropriate [A/I]
(x) Indication [1, 2, 3, 4, 5]

1. Antibiotic use as ordered is agreed with
2. Antibiotic use as ordered is agreed with, but the benefits of this therapy or prophylaxis are controversial.
3. The need for antibiotic therapy is agreed with but a different agent, or combination of agents is recommended on the basis of available clinical data, or culture of resistant organism did not result in an appropriate change in therapy
4. The need for antibiotic therapy is agreed with, but a different dose, dosing interval, route of administration or duration of therapy is recommended.
5. The need for antibiotic therapy is disagreed with, ie. there is no evidence of bacterial infection, the condition is self limiting

or prophylaxis is not indicated.

Treatment 2

When no treatment was given for numeric answers enter a zero and for choices enter "N" (not applicable)

(i)	antibiotic given	
(ii)	antibiotic analysis	[A/I/N]
(iii)	antibiotic dosage	(mg/kg/day)
(iv)	antibiotic analysis	[A/I/N]
(v)	frequency of administration	(hours)
(vi)	frequency of administration analysis	[A/I/N]
(vii)	duration of administration	(days)
(viii)	duration analysis	[A/I/N]
(ix)	route	[A/I/N]
(x)	Indication (options as for treatment 1)	[1, 2, 3, 4, 5/N]

Treatment 3

(i)	Antibiotic given	
(ii)	analysis of antibiotic given	[A/I/N]
(iii)	antibiotic dosage	(mg/kg/day)
(iv)	dosage analysis	[A/I/N]
(v)	frequency of administration	(hours)
(vi)	frequency of administration analysis	[A/I/N]
(vii)	duration of administration	(days)
(viii)	duration analysis	[A/I/N]
(ix)	route	[A/I/N]
(x)	indication (options as for treatment 1)	[1, 2, 3, 4, 5, N]

Complications

disease complications (qv table COMPLICATIONS)

WHOOPING COUGH

Clinical Features

duration cough (days)

coryzal prodrome ?
0. no 1. yes

nature of cough
2. paroxysmal 3. whoop 4. cyanotic spells
5. vomitting 6. none 7. others

Physical Signs
8. consolidation 9. scattered creps
10. rhonchi 11. none
12. others

Lab Results

white cell count
differential - neutrophils (%)
differential - lymphocytes (%)
differential - monocytes (%)
differential - eosinophils (%)
differential - basophils (%)

Chest X-ray performed [Y/N]

CXR results :-
0. not performed/no results 1. consolidation
2. collpase 3. normal

Course

died [Y/N]
duration hospital treatment (days)
duration of symptoms (days)

Contacts

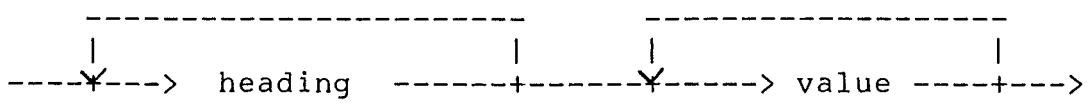
suggestive clinical history
0. none 1. parents 2. siblings
3. peers 4. medical staff 5. others

positive pernasal swabs
6. none 7. patient 8. parents
9. siblings 10. peers 11. medical staff
12. others

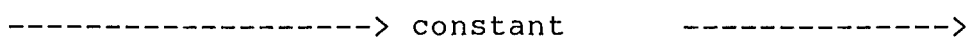
Appendix B

This appendix consists of a formal definition of PAEDIATRIC Query-By-Example.

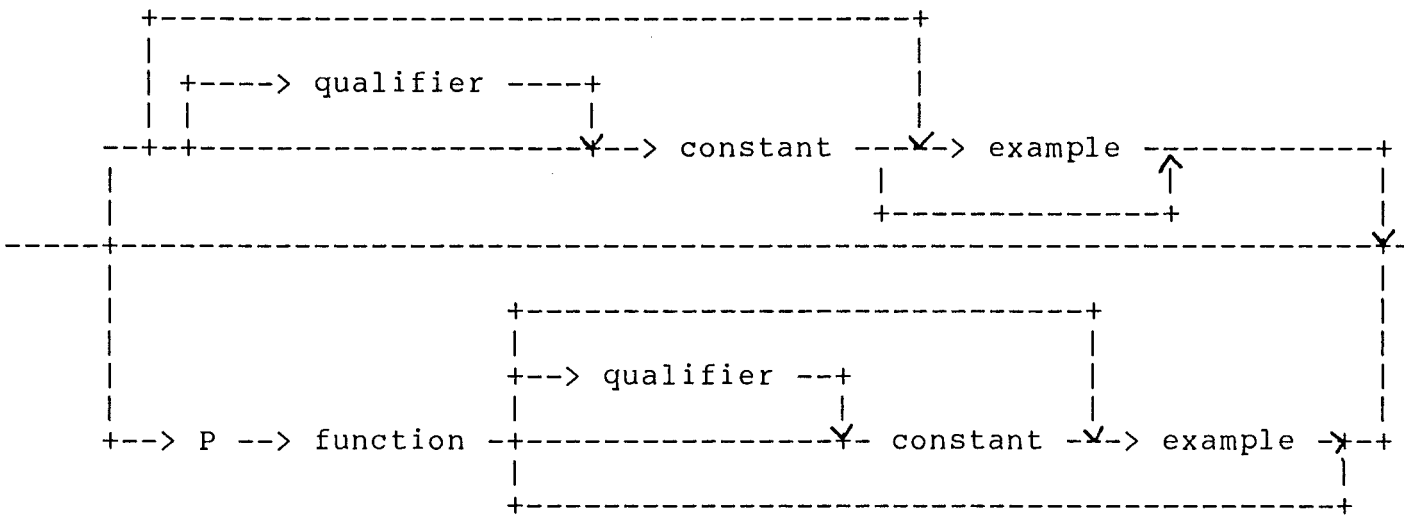
Table



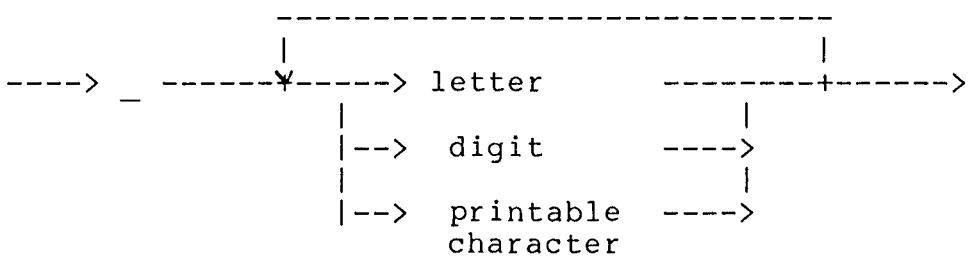
Heading



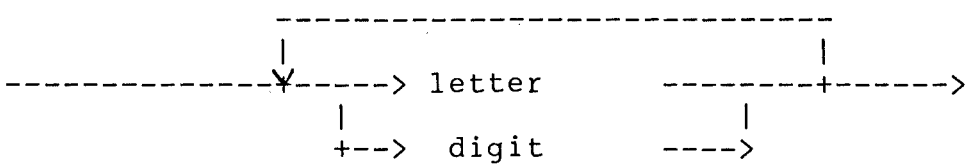
Value



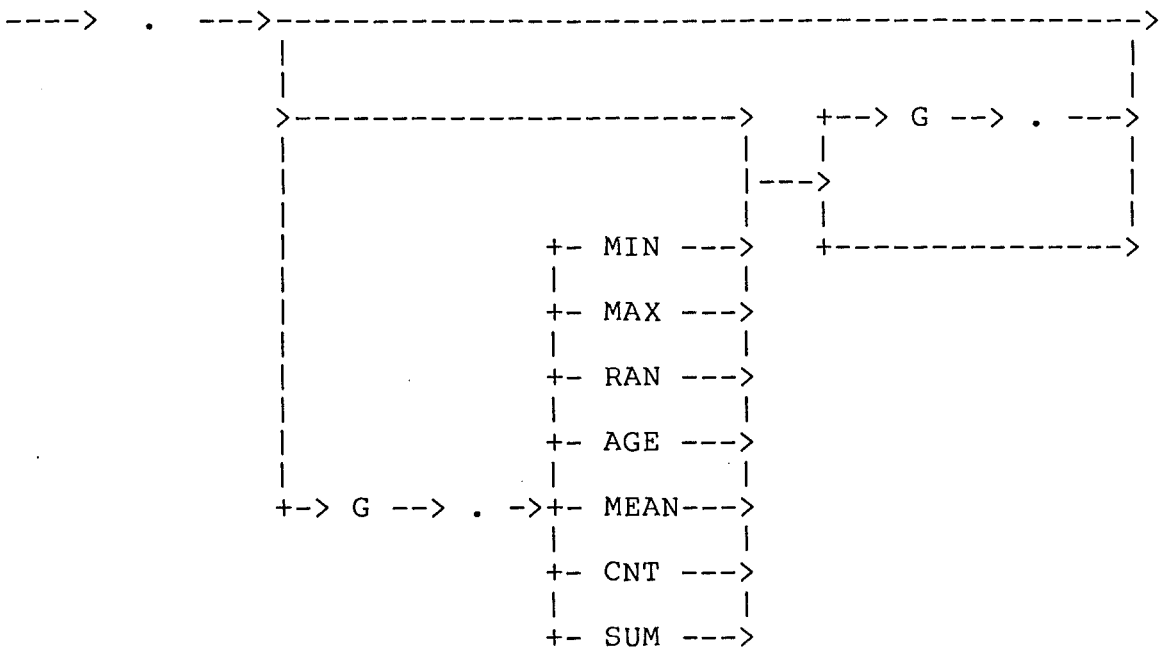
Example



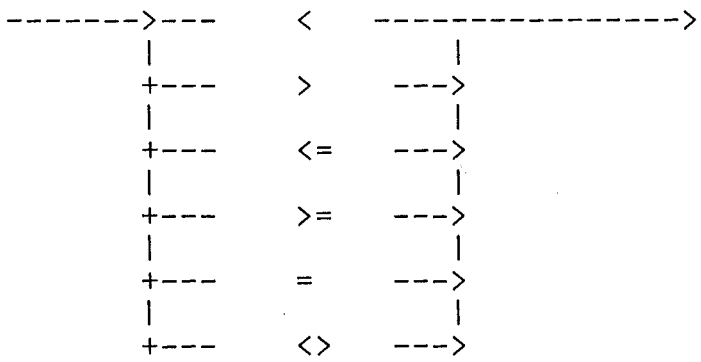
Constant



Function



Qualifier



Appendix C

This appendix consists of a set of sample queries provided by the users. They relate to the disease Whooping Cough only.

DATA RETRIEVAL QUESTIONS WITH RESPECT TO WHOOPING COUGH DURING A TIME INTERVAL LABELLED X

1. What was the number of admissions?

2. What was the age, sex, and racial distribution of the patients?

3. Was there any evidence of clustering of cases?

4. What were the social parameters of the patients admitted?

5. What was the infectious contact history -
   (a) Within the family?
   (b) Outside the family?

6. What was the immunisation status of these patients?

7. How many were receiving antimicrobial therapy before admission and what was the assessment of the appropriateness of this treatment?

8. What other relevant drugs were issued to these children before admission?

9. What was the nature and duration of the symptoms before admission?

10. What physical signs were elicited upon admission?

11. What was the range of the white cell counts and incidence of lymphocytosis?

12. What were the results of the chest x-rays performed?

13. What percentage of those admitted for whooping cough had Bordatella pertussis isolated from a nasal swab?

14. What percentage of family members had a clinical diagnosis of whooping cough?

15. What percentage of contacts with a diagnosis of whooping cough had Bordatella pertussis isolated from peri-nasal swabs?

16. What was the outcome of those children with whooping cough with respect to duration of hospitalisation, symptoms and incidence of recognised complications?

17. Were there any identifying features of children suffering from severe complications (convulsions, encephalopathy, pneumonia) or dying from whooping cough.

18. How many received erythromycin.

Ideally the data should be either displayed as the raw figures, percentages, or ~~bi~~bar graphical analysis showing the range and distribution of the data obtained.

Appendix D

Implementation Details

The tabular representation chosen was implemented to allow up to eleven domains, each with tuples to a depth of three. Syntactic error messages cause the erroneous field, with an appropriate message beneath, to be rewritten in half intensity. The message will disappear when the user types any key (the meaning of the key is lost though). The erroneous field only, need be reentered.

The number of domains and tuple depth were arbitrarily decided. The depth of the tuples was chosen to allow 23 characters/heading and 16 characters/tuple. These are the minimum widths necessary to identify each uniquely .

Regardless of the output option chosen, an output file per output table will automatically be created and the tabular results written to it. This file, at the conclusion of the retrieval, will then be written to the appropriate device and deleted as necessary.

The program performs syntax analysis as the query is entered. Reasons for choosing this method included :-

(i) avoidance of the user entering an entire query and at the conclusion receiving many confusing (and possibly contradictory) error messages

(ii) the time delay to analyse as the user enters the query is not noticeable

(iii) this method elucidates program structure.

The structure of the database is such that the input system stores information on two disks - the main and a separate disk for each disease database. The main reason for this strategy was the lack of space on the main disk to store four copies of each of the retrieval programs, storage programs, appropriate dictionaries and files. As a result of this method the situation arises of requiring random access to three disks. A workable, but not entirely satisfactory, solution which forces the user through the following steps has been devised :-

1. insert the retrieval disk in drive A:
2. insert the main disk in drive B:
3. copy the relevant file(s) (ie. the patient information and link files) to the retrieval disk
4. remove the main disk, start the retrieval program and load the appropriate disease disk.

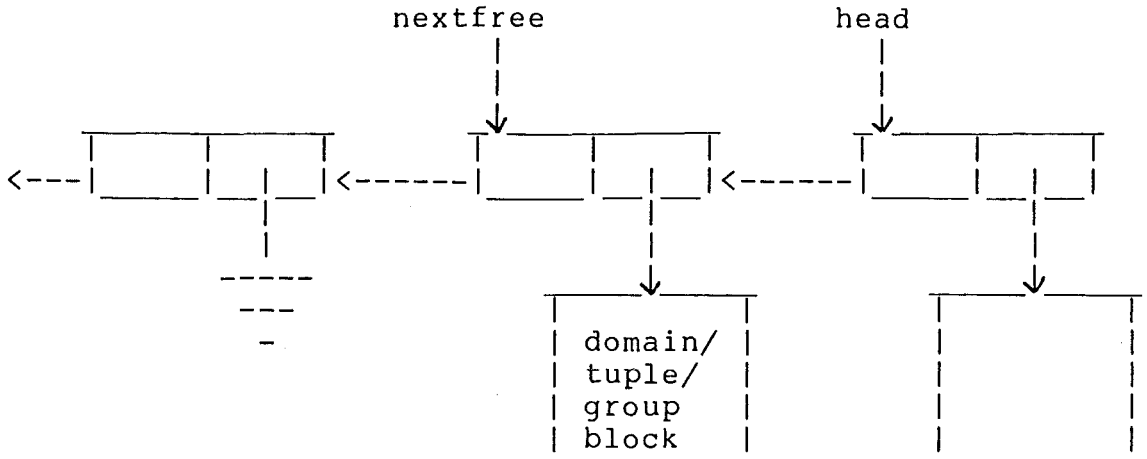
The use of a command file and the file copying utility, "PIP", would result in a relatively short execution time for this procedure.

Data Structures

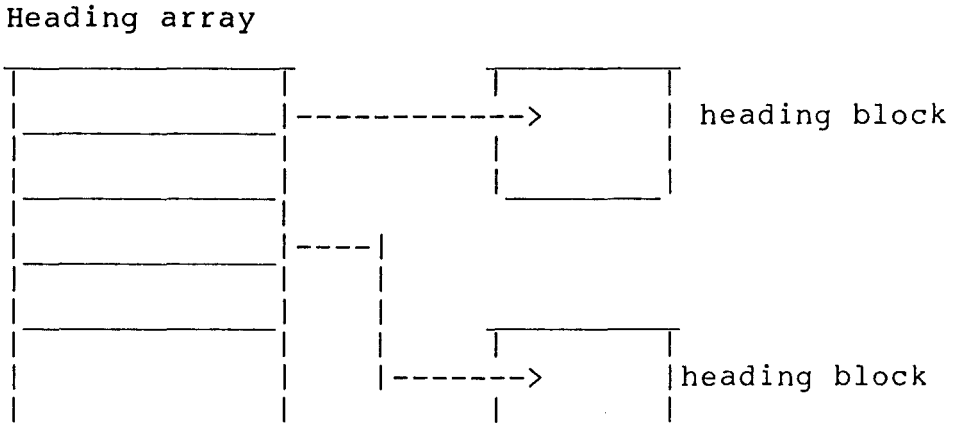
Static arrays of pointers are established for the maximum number of domains and tuples. As an attribute/tuple is created a pointer is initialised to an appropriate block.

At the time of initial entry a field is stored in a character array in a block of the appropriate type. Analysis is performed on this and meaningful information stored in the fields of the block.

Linked lists of empty records of each type are maintained in the manner illustrated below :-

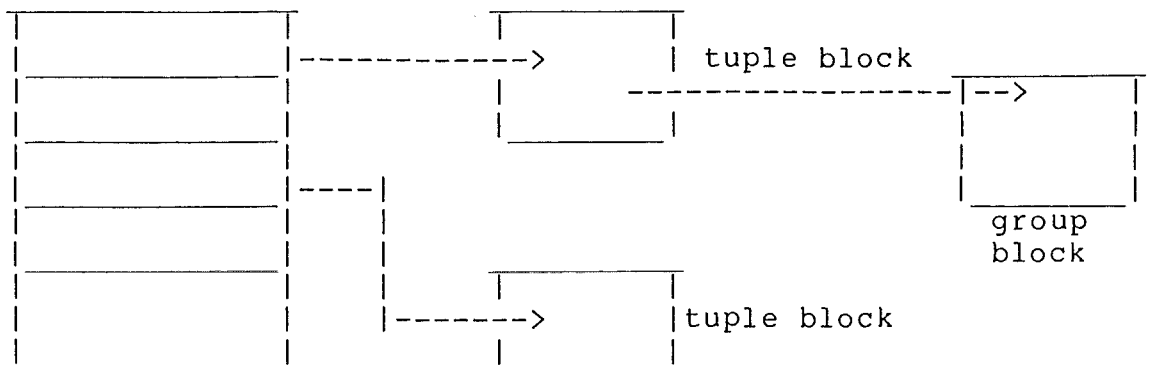


Upon erroneous entry or at the conclusion of a query the blocks are returned to the freelists.



The above diagram illustrates the manner of dynamic memory allocation for a heading. As tables allow the user unlimited freedom then a column need have no entries, thus elements without a heading block point to NIL. For a heading in the i-th column the corresponding tuple records will have indices

```
i * (maxdepth = 3 ) + 0
                    + 1
                      .
                      .
                    + (maxdepth - 1).
```



The above figure illustrates the manner of dynamic memory allocation for a tuple. Paediatric QBE offers the option of the function grouping, that is, some attributes may have logical subdivisions and the function may be applied to each subdivision. As an alternative to allocating sufficient storage for the results of the function requiring the maximum number of result variables for the maximum number of distinct groups possible, a pointer is created in the variant result record field of a tuple block. This link points to a block sufficient to hold the maximum. In a like manner to headings elements without a tuple block point to NIL. Those without a group result have some other answer type (integer, real) in the corresponding variant record field.

Each attribute must be known to the system, that is, verified by looking up (using a hashing algorithm) a dictionary. Associated with a dictionary entry is a context number, an answer type and a pointer to a group file. Thus if any tuple in a

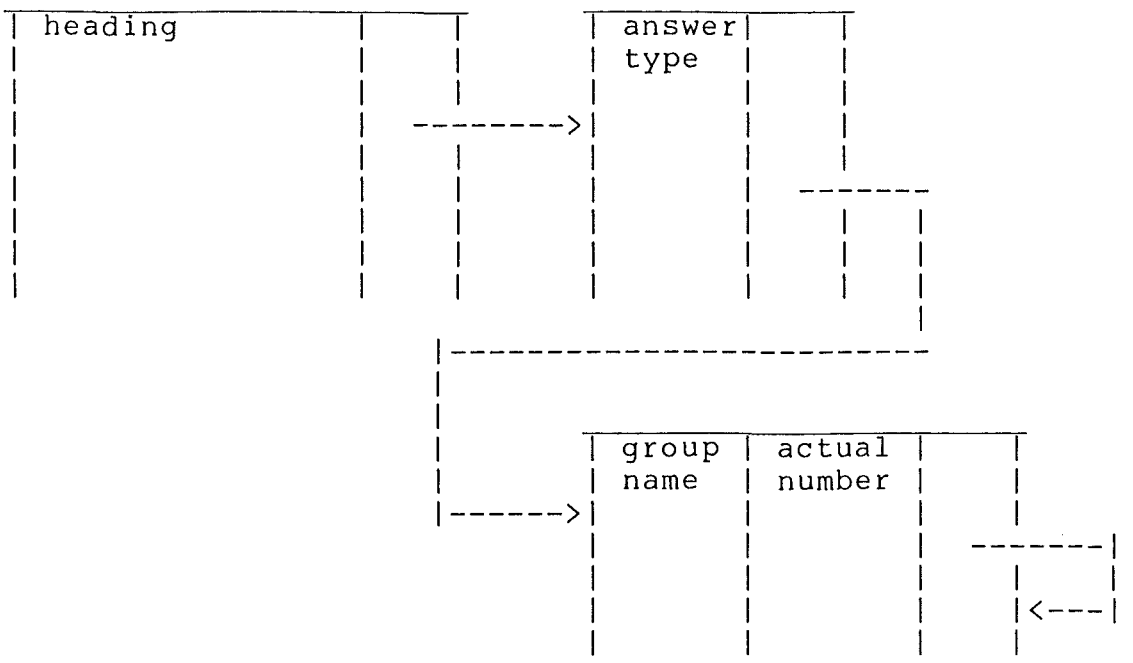
column specifies results by group then it is possible to follow a linked record structure to the groups associated with the domain. It is impossible to check the validity of numeric or character constants. However a column with an answer type indicating that its storage values were of type "datfile" (ie. the answer stored is an integer, the record number of some item in a common data file) or "set.." (ie. one or more integers representing multiple answers to a storage question were stored in a PASCAL set) means that checking must be performed for all constants specified in that column. Thus a constant in a tuple in these column types must be looked up in the tuple dictionary to check for consistency with the heading (ie. the same context number) and to obtain its storage integral value.

A tuple constant may belong to, at most, three headings. This multiple attribute definition is necessary to account for the case where a a heading may itself be a subheading and a constant will be valid for both.

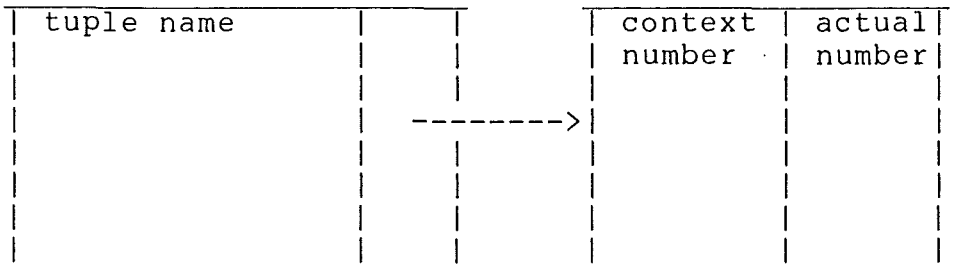
File Struture

The dictionary files may be viewed as shown over:-

Headings



Tuple files



Since there is no alteration of the context number of domains the heading dictionary is able to use this number as the pointer record number. As a consequence of a tuple's ability to belong to many domains the tuple dictionary file it is unable to employ this pointer technique. In the cases of the groups being elements of a common data file a pointer to indicate this situation is set and the name of the data file is stored.

Verification of constants is performed by the use of a "lookup" procedure using two hashing algorithms on the dictionary

files. If after hash one the record in the dictionary is occupied with a different value from that which is desired the second hash value is calculated and repeatedly added on until the word is found or the search fails.

Programs exist to create and modify these dictionary files. It was considered unnecessary at this point in the project to establish the contents of these files. Also written and tested is a program to use the hashing algorithms on the proposed contents of the dictionaries and to permit the calculation of the load factor. Clearly the time to enter all the values for this test is non-trivial and thus it has only been used with a small subset of the file contents. Before the retrieval system is put into operation different hashing functions should be tested to obtain an acceptable load factor. The chosen algorithms should then be transferred to the retrieval programs.

Retrieval

The retrieval process is illustrated by studying the following example :-

				row result
1	_x	_r	_z	true
2	_x	_s		false
3	_x	_r	_p	true
4	_y	_r		true

(i) The retrieval operation is performed by obtaining a logical result across each tuple.

(ii) A columnwise operation takes place to obtain a logical result for each column. A column is scanned for tuples

with an example. Those with the same example have their tuple results 'and'-ed with the column result and all different examples 'or'-ed.

```
column 1 : ( 1 and 2 and 3) or 4
           ( T and F and T) or T --> true
column 2 : true
column 3 : true
column 4 : true
```

(iii) Finally the column results are 'and'-ed. If the result is true the query is satisfied by the data record.

```
T and T and T and T  --> TRUE
```

To obtain the column result any tuple with a lookup constant is validated and its actual storage value obtained from the dictionary. Based on the context number of the domain a CASE statement linking the record variable name to the context number allows the comparison to take place.

```
Example :-     heading          :   Infectious contact
               answertype       :   PASCAL set 0 - 15 elements
               context #        :   10

          in table tuple :-
               constant         :   parent
          from dictionary :-
               context number   :   10
               actual #         :   1
```

CASE statement

```
CASE   casehistory^.contextno of
  .
  .
  10 : if 1 in casehistory^.contact then ....
```

Finally all the tuples are scanned and according to the result function the appropriate result field in each tuple is updated.

At the conclusion of the query all tuples are scanned and a table(s) with headings corresponding to all entries with the "P." function is created and the results output.

9. References

1. SALTON, G. "Automatic Information Organisation and Retrieval". Mc Graw - Hill Incorporated 1968.

2. DATE, C.J. "An Introduction to Database System" 3 rd ed. Addison-Wesley Publishing Company Systems Programming Series 1981.

3. "Databases : Improving Usability and Responsiveness ", edited by SHNEIDERMAN, B. Proceedings of International Conference on, Hafia, Israel 1978. New York : Academic Press

3a CODD, E.F. "How about Recently ? (English Dialogue with relational Databases using Rendezvous version 1)".

3b ZLOOF, M.M. "Design Aspects of the Query By Example Database Management Language"

3c GREENBLATT, D.
WAXMAN, J. "A Study of Three Database Query Languages"

4. "Database Management Systems, Volume 1" edited by B. SHNEIDERMAN from the National Computer Conferences, AFIPS Press 1976.

4a MYLOPOULOUS, J.
SCHUSTER, S.
TSICHRITZIS, D "A Multi-level Relational System "

4b REISNER, P.
CHAMBERLIN, D.D
BOYCE, R.F. "Human Factors Evaluation of Two Database Query Languages - SQUARE and SEQUEL".

4c ZLOOF, M.M "Query By Example"

5. Proceedings First International Confererence on Very Large Databases, edited by KERR, D.S. September, 1975

ZLOOF, M.M. "Query by Example : The Invocation and Definition of Tables and Forms ".

6. "Database Technology" Infotech State of the Art Report, Volume 2, Invited Paper 1978.

6a VANDIJCK, E.

6b PIROTTE, A.

7. THOMAS, J.C.
GOULD, J.D "A Pyschological Study of Query By Example" proceedings of the National Computer Conference, Volume 44, AFIPS Press 1975.

8. ZLOOF, M.M. I.B.M. Systems Journal Volume 16, #4 1977.

9. ULLMAN, J.D. "Principles of Database Systems". Computer Science Press 1980.

10. EVANS, C.R. "The Mighty Micro". Coronet Books, Hodder and Stoughton, 1979.

11. DEEN, S.M. "Fundamentals of Database Systems". The Macmillan Press Ltd. 1977.

12. CHEN, P. "Database Management No 6 - The Entity-Relationship Approach to Logical Database Design". MIT Solan School of Management Q.E.D. Information Series, Inc. 1977.